Design controller for synchronization of an array of delayed neural networks using a controllable probabilistic PSO☆

Yang Tang*,a,b, Zidong Wang[a,c], Jian-an Fang[a]

[a] *College of Information Science and Technology, Donghua University, Shanghai 201620, P.R. China.*
[b] *Institute of Textiles and Clothing, The Hong Kong Polytechnic University, Hong Kong, P.R. China.*
[c] *Department of Information Systems and Computing, Brunel University, Uxbridge, Middlesex, UB8 3PH, United Kingdom.*

Abstract

This paper introduces a controllable probabilistic particle swarm optimization (CPPSO) algorithm based on Bernoulli stochastic variables and a competitive penalized method. The CPPSO is proposed to solve optimization problems and is applied to design the memoryless feedback controller, which is used in synchronization of an array of delayed neural networks (DNNs). The learning strategies occur in a random way and are governed by Bernoulli stochastic variables. The expectation of Bernoulli stochastic variables are automatically controlled by search environment. The proposed method not only keeps the diversity of the swarm, but also maintains rapid convergence of the PSO according to a competitive penalized mechanism. In addition, the convergence speed is improved because the inertia weight each particle is automatically computed according to the feedback of fitness value. The efficiency of the proposed CPPSO is demonstrated by comparing it with some well-known PSO algorithms on benchmark test functions with and without rotation. In the end, the proposed CPPSO algorithm is used to design the controller for synchronization of an array of continuous-time delayed neural networks.

Key words: Swarm intelligence, neural networks, Bernoulli stochastic variable, controllable probabilistic particle swarm optimization (CPPSO), discrete and distributed delay

1. Introduction

Complex networks lie in our lives such as food webs, ecosystems, and metabolic pathways. The complexity of networks in the social, biological, engineering, and physical sciences gives rise to many challenges for scientists and engineers. As a special complex network, there has been increasing interest in artificial neural networks due to their fruitful

☆This research was partially supported by the National Natural Science Foundation of PR China (Grant No 60874113), the Research Fund for the Doctoral Program of Higher Education (Grant No 200802550007), the Key Creative Project of Shanghai Education Community (Grant No 09ZZ66), the Key Foundation Project of Shanghai(Grant No 09JC1400700), the Engineering and Physical Sciences Research Council EPSRC of the U.K. under Grant No. GR/S27658/01, an International Joint Project sponsored by the Royal Society of the U.K., and the Alexander von Humboldt Foundation of Germany.

*Corresponding author.

Email addresses: tangtany@gmail.com (Yang Tang), Zidong.Wang@brunel.ac.uk (Zidong Wang), jafang@dhu.edu.cn (Jian-an Fang)

applications in numerous areas [4, 5, 9, 13-17, 25-27, 34, 38].

Synchronization indicates that two or more systems adjust each other to lead to a common dynamical behavior. Since Pecora and Carroll proposed a method to synchronize two identical chaotic systems with different initial values [21], chaos synchronization has drawn considerable attention from various research fields such as biological networks, secure communication, and chemical reactions. Recently, arrays of coupled systems have stirred much research interest due to the fact that they can exhibit many interesting phenomena such as synchronization and autowaves, and they play a very important role in modeling interacting biological systems. Among them, the synchronization in coupled identical delayed neural networks has been found to have an important effect on the fundamental science. For instance, the synchronization hypothesis for brain activities was formulated in 1938 [35], and the modern neurophysiological experiments reinforced that synchronous oscillations of neural activities in the brain structures are important in signal processing and coding [10]. For details concerning synchronization of neural networks, we refer the authors to see [4, 5, 13, 15, 25, 26, 38], and the references cited therein. However, in these well-studied works, linear matrix inequality (LMI) and adaptive method are employed to synchronize an array of neural networks, where a number of assumptions are needed for mathematical derivation and the results obtained have conservativeness.

In search of an algorithm candidate for designing the controller for synchronization arrays of delayed NNs, the particle swarm optimization (PSO) algorithm is a competent one. The PSO has been introduced by Kennedy and Eberhart in [19] and is inspired by the motion of a flock of birds searching for food. A PSO algorithm iteratively explores a search space with a swarm of particles, searching for the global optimum. Each particle flies through the search space according to its velocity. At every step, the velocity is adjusted so that previous personal best positions and the best position found by the swarm within a specific neighborhood act as guiders. During the last decade, PSO algorithm has stirred much attention in various areas [1-3, 6-8, 11, 12, 18, 20, 22-24, 39] due to its effectiveness in performing optimization problems. Unfortunately, PSO suffers from the premature convergence problem, which does exist in complex optimization issues. In [22-24, 39], some tuning parameters methods including inertia weights and acceleration coefficients for PSOs have been proposed to enhance the PSO's search performance. A comprehensive learning PSO (CLPSO) was proposed in [12], which shows its superiority in dealing with multimodal functions. In [1, 2, 6, 8], some operators such as selection, mutation and so on have been introduced in PSOs. However, it is worth mentioning that, in almost all the works mentioned above, the convergence speed is not fast or the global search performance can be further improved. In this paper, a controllable probabilistic PSO using a competitive penalized method is developed to shorten this gap.

A CPPSO is proposed using a controllable probability method in this paper. Basically, using binary switching sequence, the PSO is switched from different learning strategies governed by Bernoulli stochastic variables. The probabilities of the stochastic variables are adjusted according to the current search information by a competitive penalized method. In addition, an adaptive inertia weight method is proposed to adjust the inertia weight automatically to improve the convergence performance. Tests are carried out on various topological structures in the PSO paradigm to demonstrate the effectiveness of the

CPPSO. Comparison between the proposed CPPSO and other improved PSO algorithms is studied comprehensively. Finally, the proposed CPPSO algorithm is used to design the controller for synchronization of an array of delayed neural networks.

2. PSO algorithms

2.1. Traditional PSO algorithms

The PSO, first introduced by Kennedy and Eberhart [19], is a stochastic optimization technique that can be likened to the behavior of a flock of birds.

In PSO, a swarm consists of N particles moving in a D-dimensional search space. The position of the ith particle is denoted by a vector, $x_i(k) = (x_{i1}(k), x_{i2}(k), \cdots, x_{iD}(k))$, where $x_{in}(k) \in [x_{\min,n}, x_{\max,n}]$ $(1 \leq n \leq D)$ with $x_{\min,n}$ and $x_{\max,n}$ being lower and upper bounds for the nth dimension, respectively. In the search process, each particle successively adjusts its own position toward the global optimum according to the two factors: the best position encountered by itself (*pbest*) denoted as $p_i = (p_{i1}, p_{i2}, \cdots, p_{iD})$ and the global best position in the whole swarm (*gbest*) denoted as $p_g = (p_{g1}, p_{g2}, \cdots, p_{gD})$. The velocity of the ith particle at the kth iteration is represented by $v_i(k) = (v_{i1}(k), v_{i2}(k), \cdots, v_{iD}(k))$, and is limited to a maximum velocity $v_{i,\max} = (v_{i\max,1}, v_{i\max,2}, \cdots, v_{i\max,D})$. $r_{1,j}$ and $r_{2,j}$ are uniform random numbers sampled from $U(0,1)$. c_1 and c_2 are the acceleration constants reflecting the weighting of stochastic acceleration terms that push the particle to *pbest* and *gbest*, respectively. The velocity v_i and position x_i of the particle at next step are updated as follows:

$$\begin{aligned} v_{i,j}(k+1) &= wv_{i,j}(k) + c_1 r_{1,j}(k)(p_{i,j}(k) - x_{i,j}(k)) + c_2 r_{2,j}(k)(p_{g,j}(k) - x_{i,j}(k)), \\ x_{i,j}(k+1) &= x_{i,j}(k) + v_{i,j}(k+1), \end{aligned} \tag{1}$$

where w is the inertia weight. In this paper, the maximum velocity $V_{\max}$ is set to the 20% of the search range [8].

2.2. Some improved PSO

PSO has attracted much attention since its introduction in 1995. Many researchers have focused on improving its search performance using various methods. One of the variants is the linearly decreasing inertia weight w introduced in [23, 24]. In [23, 24], a linearly decreased inertia weight w over time (LDIW) was proposed. Eberhart and Shi also found that the fixed or linearly decreased inertia weight for PSO is not effective for real-world applications. A random inertia weight factor was proposed when considering the dynamic nature of real-world applications [8].

On the other hand, PSO with time-varying acceleration coefficients (TVAC) was introduced by Ratnaweera *et al.* [22] . The improvement has the same motivation and the similar techniques as the LDIW, in which the cognitive coefficient c_1 is decreased linearly and the social coefficient c_2 is increased linearly over time as the follows:

$$\begin{aligned} c_1 &= (c_{1f} - c_{1i}) \times \frac{k_{\max} - k}{k_{\max}} + c_{1i}, \\ c_2 &= (c_{2f} - c_{2i}) \times \frac{k_{\max} - k}{k_{\max}} + c_{2i}, \end{aligned} \tag{2}$$

where c_{1i} and c_{2i} are the initial values of the acceleration coefficients c_1 and c_2; c_{1f} and c_{2f} are the final values of the acceleration coefficient c_1 and c_2, respectively. Usually, $c_{1i} = 2.5, c_{2i} = 0.5, c_{1f} = 0.5$ and $c_{2f} = 2.5$.

The constriction factor has been introduced into PSO for making an analysis of the convergence in [7]. It has been recommended that a constriction factor may help to realize convergence. $w = 0.729$ and $c_1 = c_2 = 2.05$ are suggested in their work. In addition, designing different kinds of structures to improve the search performance of PSO is also an hot topic. In [11], a fully informed particle swarm (FIPS) algorithm was proposed, in which the search information of the particle's entire neighborhood is used to lead the particles. Recently, some evolutionary techniques such as selection [2], crossover [6] and mutation [1] have been introduced to the PSO. On the other hand, a comprehensive-learning PSO (CLPSO) [12] was presented. Its learning method abandons the global best information, while all other particles' past best information is used to update particles's velocity instead. It is confirmed that CLPSO leads to great improvement in multimodal optimization problems. More recently, an adaptive PSO was proposed in [39]. An evolutionary factor was introduced to identify four defined evolutionary states in each generation, which enables the automatic adaption of inertia weight and acceleration coefficients.

3. A controllable probabilistic PSO

In this section, a controllable probabilistic PSO (CPPSO) is proposed for efficient search and rapid convergence speed. The learning strategies of the proposed PSO are governed by Bernoulli stochastic variables. A competitive penalized technique is introduced so as to allow the automatical control of learning probability. Furthermore, an adaptive control method for inertia weight is proposed to adjust the inertia weight of each particle. The proposed adaptive method can greatly improve the convergence speed. A local search technique is used to further refine the solution.

3.1. Control of inertia weight

The inertia weight w is usually used to balance the global and local search abilities of PSO and plays a very important role in PSO. Generally, linearly decreased inertia weight, fixed inertia weight or random inertia weight is used in variants of PSO [8, 23, 24]. It is believed that a large inertia weight enables global search and a small inertia weight facilities local search.

The fitness information characteristics of each particle vary during a PSO evolutionary process. Each particle in the swarm plays a different role in the searching process. For example, a particle which has the worst fitness may become the best particle (has best fitness) in the following generation. Thus, the fitness information of each particle would be different in searching process. Therefore, how to use this information to control the inertia weight would be a significant and promising research topic in PSO. Usually, the particle which has worst fitness should have a large velocity to fly toward the current best particle found so far or enable itself to achieve global search. The best particle should have a small speed to search around itself to refine the search solution. Based on this idea, in this paper, a novel control method of inertia weight is proposed by an evolutionary state

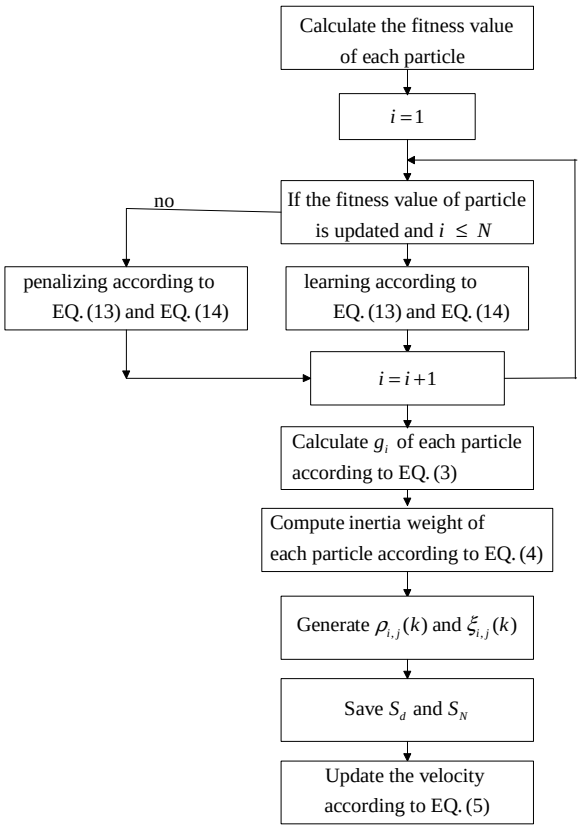


Figure 1: Flowchart of learning scheme.

function (ESF). The inertia weight is determined according to ESF automatically.

At each generation, we calculate the fitness of each particle i and get a set of fitness values $f_i (i = 1, 2, \cdots, N)$. Compare all f_i, and determine the maximum value $f_{\max}$ and minimum value $f_{\min}$. Using the set of fitness values, the ESF can be measured as

$$g_i = \frac{f_i - f_{\min}}{f_{\max} - f_{\min}}, \quad (i = 1, 2, \cdots, N) \in [0, 1]. \tag{3}$$

It is clear to see from the above equation that each particle has different ESF at each generation. The ESF can be used to determine the inertia weight of each particle. That is, the worst particle has the largest speed to enable global search and the best particle in the swarm has the smallest speed to refine the current solution. Hence, it would be beneficial to allow w to follow the evolutionary state function using the following sigmoid mapping $w(g_i) : \mathbb{R}^+ \to \mathbb{R}^+$:

$$w_i = 0.5 g_i + 0.4, \quad (i = 1, 2, \cdots, N) \in [0.4, 0.9]. \tag{4}$$

Note that by using this technique, the swarm has the inertia weight that lies in the range $[0.4, 0.9]$ in the evolutionary process. The swarm can benefit from this technique to have global search capabilities and local search capabilities at each generation simultaneously. The inertia weight of each particle will increase or decrease according to its fitness; thus, the inertia weight of each particle is time-varying in the evolutionary process. If the

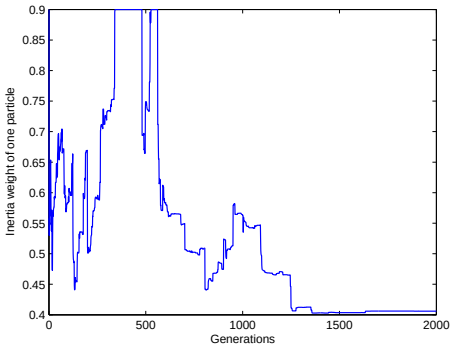


Figure 2: The parameter adaption process for inertia weight of one particle.

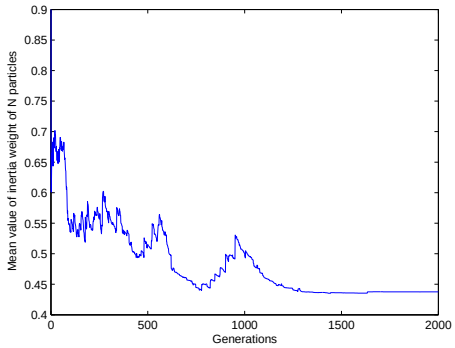


Figure 3: The parameter adaption process for mean inertia weight of the swarm.

particle has the best fitness, it will have the smallest velocity to refine itself. The worst particle (has the worst fitness) will have the largest velocity to enable a global search (learn from other particles). As w_i is monotonic with g_i, w_i will adapt to the search environment characterized by g_i. If a better solution is found outside the swarm, almost all the particles in the swarm will have large g_i and w_i, resulting in the global search. If the swarm clusters for searching solution more accurately, nearly all the particles will have small g_i and w_i, benefiting the local search.

It is worth pointing out that this technique is simple to implement in PSO since the calculation of w_i does not need extra computation. The entire process of controlling the inertia weight is illustrated in Fig. 1. An example of the change in the inertia weight of one particle and the change in the mean inertia weight of the swarm are shown in Fig. 2 and Fig. 3, respectively. The example is given for solving the Griewank function with 20 particles and 30 dimensions. It can be seen from Fig. 2 that the technique can dynamically adjust the inertia weight of the particle according to its fitness. In the early phase, the inertia weight is large to enable the particle a global search. At the end of the search process, the velocity decreases to around 0.4 to have the ability of searching locally. Fig. 3 shows that the mean value of inertia weight ranges from [0.43,0.7], a fact which describes that the state of the PSO changes into local search from global search gradually.

Remark 1. In this paper, since we define the fitness value the smaller the better when solving maximum problems, we will use negative value as the fitness value. Note that in our scheme, each particle has its own velocity belonging to $[0.4, 0.9]$. This mechanism helps the particle to play a different role in finding a better solution in the evolution process, as discussed above.

3.2. PSO with controllable probability

In this subsection, a controllable probabilistic particle swarm optimization is developed to help the PSO to achieve different learning strategies automatically. This approach is used to adaptively control the probability of different learning strategies. The proposed CPPSO combined with adaptive inertia weight is given as follows:

$$v_{i,j}(k+1) \quad = \quad w_i(k)v_{i,j}(k) + \rho_{i,j}(k)c_1r_{1,j}(k)(p_{r,j}(k) - x_{i,j}(k))$$

$$+ (1 - \rho_{i,j}(k))c_1 r_{1,j}(k)(p_{i,j}(k) - x_{i,j}(k)) + \xi_{i,j}(k)c_2 r_{2,j}(k)(p_{g,j}(k) - x_{i,j}(k)), \tag{5}$$

$$x_{i,j}(k+1) = x_{i,j}(k) + v_{i,j}(k+1), \tag{6}$$

where $p_{r,j}(k)$ stands for another particles's history best fitness in the jth dimension. $\rho_{i,j}(k)$ are stochastic variables that describe the following random events for the system (5):

$$\begin{cases} \text{Event 1:} & \text{system (5) experiences } p_{r,j}(k), \\ \text{Event 2:} & \text{system (5) experiences } p_{i,j}(k), \end{cases} \tag{7}$$

Let $\rho_{i,j}(k)$ be Bernoulli distributed sequences defined by

$$\rho_{i,j}(k) = \begin{cases} 1, & \text{if Event 1 occurs,} \\ 0, & \text{if Event 2 occurs,} \end{cases} \tag{8}$$

where $\rho_{i,j}(k)$ satisfy $\text{Prob}\{\rho_{i,j}(k) = 1\} = \rho_{i0}$,$\text{Prob}\{\rho_{i,j}(k) = 0\} = 1 - \rho_{i0}$.

Similarly, $\xi_{i,j}(k)$ are stochastic variables that describe the following random events for the system (5):

$$\begin{cases} \text{Event 1:} & \text{system (5) experiences } p_{g,j}(k), \\ \text{Event 2:} & \text{system (5) does not experience } p_{g,j}(k), \end{cases} \tag{9}$$

Set $\xi_{i,j}(k)$ be Bernoulli distributed sequences defined by

$$\xi_{i,j}(k) = \begin{cases} 1, & \text{if Event 1 occurs,} \\ 0, & \text{if Event 2 occurs,} \end{cases} \tag{10}$$

where $\xi_{i,j}(k)$ satisfy $\text{Prob}\{\xi_{i,j}(k) = 1\} = \xi_0$, $\text{Prob}\{\xi_{i,j}(k) = 0\} = 1 - \xi_0$.

In summary, system (5) will experience four cases according to the different stochastic variables $\rho_{i,j}(k)$ and $\xi_{i,j}(k)$. The state of system (5) is determined as follows:

$$S = \begin{cases} 1, & \text{if } \rho_{i,j}(k) = 1 \text{ and } \xi_{i,j}(k) = 0, \\ 2, & \text{if } \rho_{i,j}(k) = 1 \text{ and } \xi_{i,j}(k) = 1, \\ 3, & \text{if } \rho_{i,j}(k) = 0 \text{ and } \xi_{i,j}(k) = 0, \\ 4, & \text{if } \rho_{i,j}(k) = 0 \text{ and } \xi_{i,j}(k) = 1. \end{cases} \tag{11}$$

Obviously, if $S = 1$ or $S = 3$, the learning strategy of system (5) becomes the learning scheme proposed in [12]. If $S = 4$, the learning strategy of system (5) turns into the original PSO [19]. In our proposed system (5), the term $p_{g,j}(k)$ is added to enhance the local search performance. In [12], it was found that comprehensive learning techniques will improve the global search performance of PSO. However, the convergence speed is slow and the selection of learning probability is empirically rather than intelligently. Note that different learning probability values ρ_{i0} yield different search performance in multimodal functions. Similarly, different ξ_0 offers various results of PSO when dealing with different functions. Thus, a competitive penalized method is proposed here to balance the tradeoff between local search and global search abilities.

Remark 2. Note that we introduce the stochastic variables $\rho_{i,j}(k)$ and $\xi_{i,j}(k)$ to describe different learning strategies occurring in a random way. Using binary sequence switching to describe stochastic event is widely used in our previous works, such as miss measurement, random packet losses and stochastic delay [28, 31-33].

Remark 3. The stochastic variables used in our paper can well describe the loss information conveyed in a flock of birds, forming the original idea from which PSO arises. The intermittent information losses are known in the process of communication among the particles in the swarm. In this paper, the information losses are viewed as a binary switching sequence that is specified by a conditional probability distribution.

Remark 4. Clearly, the learning strategies in our proposed model can cover the learning scheme proposed in [12] as a special case. If $S = 1$ or $S = 3$, the learning strategy of system (5) becomes the learning scheme proposed in [12]. *gbest* is also added in our model to improve the convergence speed. In addition, a competitive penalized approach is proposed in this paper to select efficient learning strategy and abandon inefficient learning strategy instead of using fixed learning strategy.

For each dimension of the particle i, two random numbers are generated by sampling from $[0, 1]$. Determine the state of system (5) using two random numbers. As shown in (11), we can further summarize the learning strategies as follows:

$$\begin{cases} \text{learn from } p_{r,j}(k), & \text{if } S = 1, \\ \text{learn from } p_{r,j}(k) \text{ and } p_{g,j}(k), & \text{if } S = 2, \\ \text{learn from } p_{i,j}(k), & \text{if } S = 3, \\ \text{learn from } p_{i,j}(k) \text{ and } p_{g,j}(k), & \text{if } S = 4. \end{cases} \tag{12}$$

We employ the selection procedure of the particles's dimension learning strategies as follows.

(1) Choose the learning strategy at each generation according to learning probability ρ_{i0} and ξ_0.

(2) If the learning strategy is $S = 1$ or $S = 3$, randomly choose three particles from the population. Compare the fitness values of these three particles's $p_i(k)$ and select the best one. Use the winner's *pbest* as the exemplar to learn from that dimension. Save the learning strategy of each dimension as $S_d(j)(i = 1, 2, \cdots, N * D)$. If the learning strategy is $S = 2$ or $S = 4$, update the velocity of the dimension according to (5).

(3) After all the dimensions in each particle have learned from one of the four strategies, we randomly generate an integer number q ranges from $[1, N * D]$. Using this random number, we get the each particle's learning strategy $S_N = S_d(q)(i = 1, 2, \cdots, N)$.

(4) After the velocities of all the particles have been updated, the activation probability variables ρ_{i0} are updated as follows:

$$\rho_{i0} = \begin{cases} \rho_{i0} + \alpha, & \text{if the fitness value of particle } i \text{ is updated} \\ & \text{and } S_N = 1 \text{ or } S_N = 2, \\ \rho_{i0} - \beta, & \text{if the fitness value of particle } i \text{ is not updated} \\ & \text{and } S_N = 3 \text{ or } S_N = 4, \end{cases} \tag{13}$$

where α is the learning rate and β is the penalizing rate.

(5) Similarly, the activation probability variable ξ_0 is updated according to:

$$\xi_0 = \begin{cases} \xi_0 + \alpha, & \text{if the fitness value of particle } i \text{ is updated} \\ & \text{and } S_N = 2 \text{ or } S_N = 4, \\ \xi_0 - \alpha, & \text{if the fitness value of particle } i \text{ is not updated} \\ & \text{and } S_N = 2 \text{ or } S_N = 4, \end{cases} \tag{14}$$

In addition to the control method of inertia weight, we observe from the learning method that there exist three differences between the learning strategies CLPSO and the proposed learning methods.

(1) Instead of using particle's own $p_i(k)$ and its neighbor's $p_r(k)$, we also employ the $p_g(k)$ to guide the particle to fly to the current global optima found so far quickly.

(2) Instead of using two learning strategies to control PSO, we use four learning strategies to control PSO.

(3) The activation probability variables ρ_{i0} and ξ_0 are automatically adjusted by the current search information. An efficient competitive penalized approach is proposed here to adapt the variables to the appropriate values.

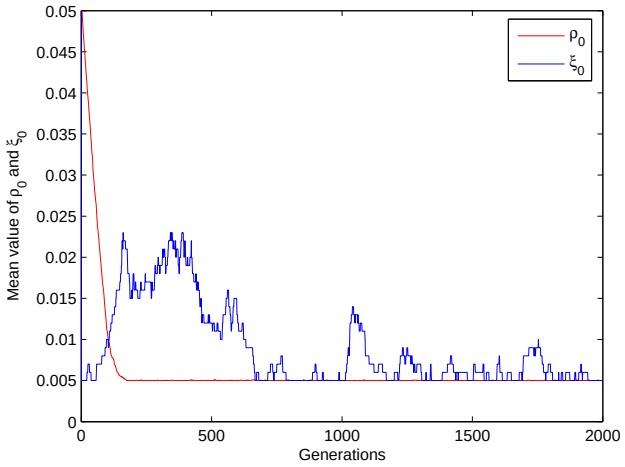


Figure 4: The parameter adaption process for ρ_0 and xi_0.

Note that all the particles have the same ξ_0, which can enable the quick convergence speed or enlarge the search diversity of the swarm rapidly. ρ_{i0} are initialized to 0.05 and adaptively adjusted according to the learning rate α and penalizing rate β. ξ_0 is initialized to 0.005 since the swarm benefits the global search in the early phase. If ρ_{i0} or ξ_0 are smaller than 0.005, we set $\rho_{i0} = 0.005$ or $\xi_0 = 0.005$, respectively. $S_N(i)$ is initialized to 1 throughout this paper. The flowchart of this technique can be seen from Fig. 1. CPPSO with the method proposed in subsection 3.1 and 3.2 is named CPPSO-I in this paper. An example of the change in mean value of ρ_{i0} and ξ_0 of the swarm is shown in Fig. 4. It is for solving the Griewank function with 20 particles and 30 dimensions. It can be seen from Fig. 4 that the technique can dynamically adjust ρ_{i0} and ξ_0 to suitable values by a

competitive penalized method. From Fig. 4, it is observed that the ρ_{i0} should keep a low value to achieve better solution when tackling with Griewank function.

Remark 5. The competitive penalized method proposed in our paper can be summarized as follows. If the current learning strategy is efficient to find the global optimum, that is, the current learning strategy can update the fitness value successfully, then the learning probability will increase in order to promote more frequent occurrence. If the learning strategy employed is useless to update the fitness value, the learning method will be penalized by reducing the learning probability.

3.3. Elite local learning approach

In this subsection, an elite local learning approach (ELLA) is used to accelerate the convergence speed. This approach is controlled automatically in the evolutionary process. Let $\gamma(k)(k \geq 0)$ be a Markov chain taking values in a finite state space $\mathcal{S} = \{1, 2, \cdots, N\}$ with probability transition matrix $\Gamma^{(k)} = (\alpha_{ij}^{(k)})_{N\times N}$ given by

$$\mathcal{P} = \{\gamma(k+1) = j | \gamma(k) = i\} = \alpha_{ij}^{(k)}, i, j = 1, 2, \cdots, N, \tag{15}$$

where $\alpha_{ij}^{(k)} \geq 0 (i, j \in \mathcal{S})$ is the transition rate from i to j and $\sum_{j=1}^{N} \alpha_{ij}^{(k)} = 1$.

The ELLA randomly selects one dimension of the globally best particle, which is represented by P_j for the jth dimension. Only one dimension is chosen in that the local optima are likely to have better solution in one dimension. Each dimension of the globally best particle has the same probability to be chosen. The elite learning is carried out by the following two search strategies

$$P_j = P_j + ((\lambda_{1,j} - \lambda_{2,j}) * r_1 - \lambda_{1,j}) * R_1, \tag{16}$$

$$P_j = P_j + ((\eta_1 - \eta_2) * r_2 - \eta_1) * R_2, \tag{17}$$

where r_1 and r_2 are uniformly distributed random numbers sampled from $U(0, 1)$. $\lambda_{1,j}$ and $\lambda_{2,j}$ are the upper and lower bounds of the chosen particle in the swarm in jth dimension at each step. η_1 and η_2 are the maximal and minimal value of all dimensions of the globally best particle at each generation. R_1 and R_2 are the search radii of the learning method which are controlled by a probability matrix $\Gamma^{(k)}$.

To choose the R_1 and R_2, consider the following probability transition matrix

$$\Gamma^{(k)} = \begin{pmatrix} \alpha^{(k)} & 1 - \alpha^{(k)} \\ \alpha^{(k)} & 1 - \alpha^{(k)} \end{pmatrix}, \tag{18}$$

where $\alpha^{(k)}$ is time-varying and is used to choose the radius. $\gamma(k) = 1$ denotes that the $R_i (i = 1, 2)$ is taken as $\rho_1 = 1$, while $\gamma(k) = 2$ indicates that the R_i is chosen as $\rho_2 = 0.1$. It is suggested that $\alpha^{(k)}$ should decrease linearly with time and is given by

$$\alpha^{(k)} = (\alpha_1 - \alpha_2) \times \frac{k_{\max} - k}{k_{\max}} + \alpha_2, \tag{19}$$

where α_1 and α_2 are the upper and lower bounds of $\alpha^{(k)}$. We set $\alpha_1 = 1$ and $\alpha_2 = 0$ in this paper, which indicates that ρ_1 occurs frequently and ρ_2 seldom happens in the early stage. In elite learning, the new position is accepted if the fitness is better than the current globally best position. Otherwise, the new position is abandoned.

One problem that should be taken into consideration is that these two search methods are not always useful in the search process. A competitive penalized learning method is proposed here to deal with this problem. Define Q, Q_1 and Q_2 as ELLA activation variable, local search learning (16) activation variable and local search learning (17) activation variable, respectively. We make use of Q to control the activation of ELLA. Meantime, Q_1 and Q_2 are used to activate (16) and (17), respectively. Q is updated as follows:

$$Q = \begin{cases} Q + \theta, & \text{if one of the search strategy } i \\ & \text{succeed to update the globally best particle,} \\ Q, & \text{if the search strategy } i \\ & \text{fails to improve the globally best particle,} \end{cases} \tag{20}$$

where θ is the learning rate, which is fixed as $\theta = 0.05$ in this paper.

The activation probability variables Q_1 and Q_2 are updated as follows:

$$Q_i = \begin{cases} Q_i + \delta_1, & \text{if the search strategy } i \\ & \text{succeed to update the globally best particle,} \\ Q_i - \delta_2, & \text{if the search strategy } i \\ & \text{fails to improve the globally best particle,} \end{cases} \tag{21}$$

where $i = 1, 2$ denotes the search strategy (16) and (17), respectively. δ_1 is the learning rate and δ_2 is the penalizing rate. In this paper, we fix $\delta_1 = 0.5$ and $\delta_2 = 0.001$. If the magnitude of the updated $Q_i (i = 1, 2)$ exceeds 1, then Q_i is assigned the value 1. Similarly, if Q_i decreases under 0.01, then Q_i is set to 0.01. CPPSO with the method proposed in subsection 3.1, 3.2 and 3.3 is named CPPSO-II in this paper.

In summary, the pseudo code of CPPSO-II algorithm is described as follows by above discussion:

```
CPPSO.InitializeParameters();
while (FE is not equal to FE_max)
{
for i = 1:particle numbers N
    {
    CPPSO.UpdatePosition();// update positions of particle i according to Eq. (6)
    val=CPPSO.CalculateFitness();// calculate fitness of particle i
    CPPSO.UpdateFE();// calculate FEs
        if (val < the best fitness of particle i found so far)
        {
        save the position x_i(k) and val;
        CPPSO.UpdateLearnProbability();// according to Eq. (13) and Eq. (14)
        }
        else CPPSO.PenalizeLearnProbability();// according to Eq. (13) and Eq. (14)
    }
```

```
if (Rand() < Q)
    {
    if (Rand()< Q_1)
        {
        CPPSO.GenerateRadius();// according to Eq. (15), (18) and (19)
        CPPSO.ELLA_1();// according to Eq. (16)
        val=CPPSO.CalculateFitness();
        CPPSO.UpdateFE();
        if (val < the best fitness in the swarm found so far)
            {
            save the position and val to the best particle;
            CPPSO.RewardQ();// according to Eq. (20)
            CPPSO.RewardQ_1();// according to Eq. (21)
            }
        else CPPSO.PenalizeQ_1();// according to Eq. (21)
        }
    if (Rand()< Q_2)
        {
        CPPSO.GenerateRadius();// according to Eq. (15), (18) and (19)
        CPPSO.ELLA_2();// according to Eq. (17)
        val=CPPSO.CalculateFitness();
        CPPSO.UpdateFE();
        if (val < the best fitness in the swarm found so far)
            {
            save the position to best particle and val;
            CPPSO.RewardQ();// according to Eq. (20)
            CPPSO.RewardQ_2();// according to Eq. (21)
            }
        else CPPSO.PenalizeQ_2();// according to Eq. (21)
        }
    }
for i = 1:particle numbers N
    {
    CPPSO.CalculateWeight();// calculate each particle's weight using Eq. (3) and (4)
    for j = 1:dimension D
        {
        CPPSO.UpdateVelocity();// update velocity according to Eq. (5), ρ_i0 and ξ_0
        }
    }
    CPPSO.SaveLearnStrategy();// save S_N
}
```

4. Experiments

In the experiments, twelve well-known benchmarks [30, 37] have been used to test the performances of CPPSO. The experiments are performed to verify the effectiveness of SPSO and compare the CPPSO with other well-known PSOs to show its superiority.

4.1. Experiments setup

Twelve benchmark functions are listed in Table 1 and (22)-(33) are used to test the performance of PSOs. All the functions are tested on 30 dimensions. The population sizes of all the PSOs are set to 20. $f_1(x)$ and $f_2(x)$ are unimodal optimization problems. $f_1(x)$ is used to test the convergence speeds of PSOs. $f_2(x)$ can be treated as a multimodal problem since it has a narrow valley from the perceived local optima to the global optimum. $f_3(x)$ to $f_8(x)$ are multimodal problems which are hard to optimize. Note that some functions are separable and can be solved by using D one-dimensional searches. Hence, four rotated multimodal problems are used to test the performance of the PSOs. To rotate a function, first an orthogonal matrix M should be generated according to the Salomon's method [29]. The original variable x is left multiplied by the orthogonal matrix M to get the new rotated variable $y = M * x$. This variable y is used to compute the fitness value f. Clearly, when one dimension in x is changed, all dimensions in y will be influenced. Thus, the rotated function cannot be solved by D one-dimensional searches.

$$\text{Sphere} : f_1(x) = \sum_{i=1}^{D} x_i^2, \tag{22}$$

$$\text{Rosenbrock} : f_2(x) = \sum_{i=1}^{D-1} (100(x_{i+1} - x_i)^2 + (x_i - 1)^2), \tag{23}$$

$$\text{Weierstrass} : \quad f_3(x) = \sum_{i=1}^{D} \left(\sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k (y_i + 0.5))] - D \sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k * 0.5)] \right), \quad a = 0.5, b = 3, k_{\max} = 20, \tag{24}$$

$$\text{Rastrigin} : f_4(x) = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10), \tag{25}$$

$$\text{Noncontinuous Rastrigin} : \quad f_5(x) = \sum_{i=1}^{D} (y_i^2 - 10\cos(2\pi y_i) + 10), \quad \text{where} \quad y_i = \begin{cases} x_i, & |x_i| < 0.5, \\ \frac{round(2x_i)}{2}, & |x_i| \geq 0.5, \end{cases} \tag{26}$$

$$\text{Ackley} : f_6(x) = -20e^{-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}} - e^{\frac{1}{D}\sum_{i=1}^{D} \cos 2\pi x_i} + 20 + e, \tag{27}$$

$$\text{Griewank}: f_7(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}}) + 1, \tag{28}$$

Generalized Penalized :

$$f_8(x) = \frac{\pi}{D}\{10\sin^2(\pi y_1) + \sum_{i=1}^{D-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})] + (y_D-1)^2\}$$
$$+\sum_{i=1}^{D} u(x_i, 10, 100, 4),$$
$$\text{where} \quad y_i = (1+\frac{1}{4}(x_i+1)), u(x_i,a,k,m) = \begin{cases} k(x_i-a)^m, & x_i > a, \\ 0, & -a \le x_i \le a, \\ k(-x_i-a)^m, & x_i < -a. \end{cases} \tag{29}$$

$$\text{Rotated Rastrigin}: f_9(x) = \sum_{i=1}^{D}(y_i^2 - 10\cos(2\pi y_i) + 10), \qquad y = M * x, \tag{30}$$

Rotated noncontinuous Rastrigin :

$$f_{10}(x) = \sum_{i=1}^{D}(z_i^2 - 10\cos(2\pi z_i) + 10),$$
$$\text{where} \quad z_i = \begin{cases} y_i, & |y_i| < 0.5, \\ \frac{round(2y_i)}{2}, & |y_i| \ge 0.5, \end{cases} \qquad y = M * x, \tag{31}$$

$$\text{Rotated Griewank}: f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{D} y_i^2 - \prod_{i=1}^{D}\cos(\frac{y_i}{\sqrt{i}}) + 1, \qquad y = M * x, \tag{32}$$

$$\text{Rotated Rosenbrock}: f_{12}(x) = \sum_{i=1}^{D-1}(100(y_{i+1}-y_i)^2 + (y_i-1)^2), \qquad y = M * x, \tag{33}$$

Experiments are conducted to compare six PSO algorithms including the proposed CPPSO-I and CPPSO-II on the 12 test problems with 30 dimensions. The population size of six PSO is 20. Four existing PSO algorithms are shown in Table 2 in detail. The first PSO is LDIW [23, 24] with linearly decreasing inertia weight. TVAC [22] is a PSO with time-varying acceleration parameters and incorporating a self-organizing method. CLPSO delivers a comprehensive-learning strategy, which is used to yield better performance for multimodal functions [12]. APSO is an adaptive PSO, which can adjust the acceleration coefficients and inertia weight adaptively [39] according to an evolutionary factor by calculating average distance. The parameters for these PSOs are provided in Table 2.

Table 1: Benchmark configurations

Functions	Name	Dimension	Search Space	Minimum	Threshold
$f_1(x)$	Sphere	30	$[-100, 100]^D$	0	0.01
$f_2(x)$	Rosenbrock	30	$[-10, 10]^D$	0	100
$f_3(x)$	Weierstrass	30	$[-0.5, 0.5]^D$	0	0.01
$f_4(x)$	Rastrigin	30	$[-5.12, 5.12]^D$	0	50
$f_5(x)$	Noncontinuous Rastrigin	30	$[-5.12, 5.12]^D$	0	50
$f_6(x)$	Ackley	30	$[-32, 32]$	0	0.01
$f_7(x)$	Griewank	30	$[-600, 600]^D$	0	0.01
$f_8(x)$	Generalized Penalized	30	$[-50, 50]^D$	0	0.01
$f_9(x)$	Rotated Rastrigin	30	$[-5.12, 5.12]^D$	0	50
$f_{10}(x)$	Rotated noncontinuous Rastrigin	30	$[-5.12, 5.12]^D$	0	50
$f_{11}(x)$	Rotated Griewank	30	$[-600, 600]^D$	0	0.01
$f_{12}(x)$	Rotated Rosenbrock	30	$[-10, 10]^D$	0	100

Table 2: PSO algorithms for comparison

Algorithm	Parameters	Reference
LDIW	$w : 0.9 - 0.4, c_1 = c_2 = 2$	[23]
TVAC	$w : 0.9 - 0.4, c_1 : 2.5 - 0.5, c_2 : 0.5 - 2.5$	[22]
CLPSO	$w : 0.9 - 0.4, c = 1.49, m = 7$	[12]
APSO	Automatically chosen	[39]

In all the experiments, the algorithm configuration of the CPPSO is listed as follows. Learning rate α and penalizing rate β are set to 0.001 and 0.001, respectively. Search radii ρ_1 and ρ_2 are chosen as 1 and 0.1, respectively. The initial states are all set to $S_N(i) = 1(i = 1, 2, \cdots, N)$.

All the algorithms use the same number of 2×10^5 fitness evaluations (FEs) for each test function, as suggested in [30]. Further, all the experiments are performed on the same machine with a Core 2 2.26-GHz CPU, 2-GB memory, and Windows XP operating system. Each algorithm will repeat 30 times independently for eliminating random discrepancy.

4.2. Adjust learning rate and penalizing rate

In this subsection, the effect of learning rate α and penalizing rate β are investigated on CPPSO-I here. For the sake of simplicity, in this paper, we assume $\alpha = \beta$. Appropriate α and β can enhance global and local search capabilities and reduce inefficient learning strategy, leading to saving the consumption of FEs. The results of mean values and standard deviations of the solutions are provided in Table 1. α is fixed to 0.0001, 0.0005, 0.001, 0.005 and 0.01 in the experiment, respectively.

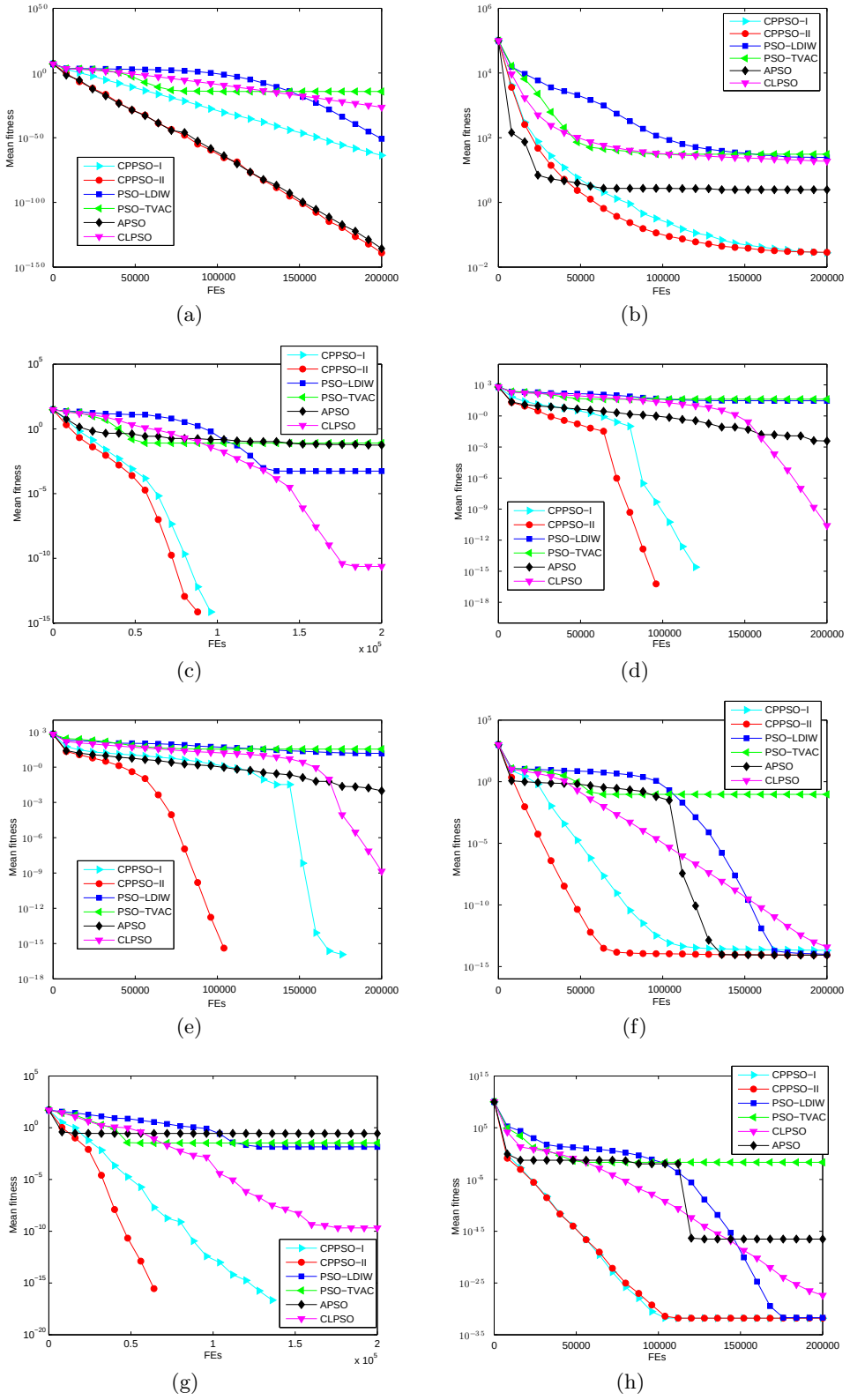
From the results, it can be found that $\alpha = \beta = 0.001$ reveals the best performance. Large α and β give rise to new learning strategy quickly, while small α and β lead to slow change the learning strategy. Therefore, in this paper, $\alpha = \beta = 0.001$ is adopted.

Table 3: Effects of the learning rate and penalizing rate on search accuracy and convergence rate(The best value is in Bold font.)

	α	0.0001	0.0005	0.001	0.005	0.01
f_1	Average	2.5×10^{-58}	6.2×10^{-60}	$\mathbf{1.4}\times10^{-63}$	1.3×10^{-61}	4.2×10^{-5}
	Mean FEs	19172	21783	21735	19377	**19110**
f_2	Average	11.6	0.65	**0.02**	1.07	11
	Mean FEs	**12479**	21687	21641	22033	24032
f_4	Average	0.06	0.06	**0**	**0**	**0**
	Mean FEs	19408	10927	**9451**	11038	11508
f_6	Average	$\mathbf{2.0}\times10^{-14}$	2.1×10^{-14}	2.4×10^{-14}	0.0001	0.004
	Mean FEs	15888	28113	**13801**	32331	32742
f_9	Average	60.1224	50.7438	**37.9**	70.6456	68.6524
	Mean FEs	-	67232	**43551**	-	-

Table 4: Search result comparisons among six PSOs on twelve test functions

		LDIW	TVAC	CLPSO	APSO	CPPSO-I	CPPSO-II
f_1	Mean	2.3×10^{-50}	2.6×10^{-17}	2.5×10^{-27}	4.2×10^{-132}	1.4×10^{-63}	$\mathbf{2.9}\times10^{-140}$
	Best Value	1.2×10^{-55}	3.2×10^{-32}	2.1×10^{-28}	3.9×10^{-148}	1.7×10^{-72}	$\mathbf{9.9}\times10^{-159}$
	Std. Dev.	4.3×10^{-50}	6.4×10^{-19}	2.3×10^{-27}	5.6×10^{-145}	2.3×10^{-64}	$\mathbf{1.6}\times10^{-138}$
f_2	Mean	29.7	31.8	18.7	2.1	**0.02**	**0.02**
	Best Value	3.8	2.4	9.7	0.0006	**0.0001**	**0.0001**
	Std. Dev.	37.0	25.7	5.4	1.1	0.04	**0.03**
f_3	Mean	2.1×10^{-4}	0.0801	2.3×10^{-11}	0.05	**0**	**0**
	Best Value	5.6×10^{-4}	0.002	4.2×10^{-12}	0.003	**0**	**0**
	Std. Dev.	3.7×10^{-4}	0.02	1.7×10^{-11}	0.03	**0**	**0**
f_4	Mean	25.8	47.9	3.7×10^{-11}	0.003	**0**	**0**
	Best Value	13.6	23.8	**0**	**0**	**0**	**0**
	Std. Dev.	25.1	11.6	1.5×10^{-10}	0.007	**0**	**0**
f_5	Mean	14.8	34.1	1.1×10^{-9}	0.003	**0**	**0**
	Best Value	7.9	16.2	2.3×10^{-11}	1.2×10^{-15}	**0**	**0**
	Std. Dev.	14.3	15.7	2.2×10^{-9}	0.004	**0**	**0**
f_6	Mean	9.7×10^{-14}	0.09	3.7×10^{-14}	$\mathbf{8.4}\times10^{-15}$	2.4×10^{-14}	1.3×10^{-14}
	Best Value	7.7×10^{-15}	2.8×10^{-13}	2.5×10^{-14}	$\mathbf{4.1}\times10^{-15}$	$\mathbf{4.1}\times10^{-15}$	7.7×10^{-15}
	Std. Dev.	7.4×10^{-14}	0.3	6.7×10^{-14}	$\mathbf{2.5}\times10^{-15}$	1.7×10^{-14}	1.9×10^{-15}
f_7	Mean	0.01	0.03	2.7×10^{-11}	0.0008	**0**	**0**
	Best Value	**0**	**0**	4.4×10^{-12}	0.0001	**0**	**0**
	Std. Dev.	0.02	0.01	1.2×10^{-11}	0.001	**0**	**0**
f_8	Mean	1.9×10^{-32}	0.02	4.1×10^{-28}	3.0×10^{-17}	$\mathbf{1.5}\times10^{-32}$	$\mathbf{1.5}\times10^{-32}$
	Best Value	$\mathbf{1.5}\times10^{-32}$	1.2×10^{-26}	3.0×10^{-17}	1.5×10^{-32}	$\mathbf{1.5}\times10^{-32}$	$\mathbf{1.5}\times10^{-32}$
	Std. Dev.	3.7×10^{-33}	0.04	5.6×10^{-29}	1.41×10^{-26}	**0**	**0**
f_9	Mean	53.3	50.2	43.3	200.0	**37.9**	41.9
	Best Value	30.8	32	35.4	89.2	**20.8**	33.2
	Std. Dev.	20.4	14.8	6.7	152.3	8.5	**5.3**
f_{10}	Mean	71.4	59.4	49.7	176.2	46.8	**41.8**
	Best Value	43	27	38.1	97	35.2	**31.8**
	Std. Dev.	12.0	16.5	**5.5**	48.6	9.3	6.0
f_{11}	Mean	0.0176	0.02	1.7×10^{-6}	0.0113	5.5×10^{-10}	$\mathbf{1.2}\times10^{-12}$
	Best Value	**0**	5.5×10^{-15}	2.3×10^{-8}	9.0×10^{-6}	**0**	**0**
	Std. Dev.	0.02	0.01	2.3×10^{-6}	0.0088	7.3×10^{-10}	$\mathbf{3.1}\times10^{-12}$
f_{12}	Mean	48.6	37.6	25.1	51.0	22.3	**20.5**
	Best Value	16.4	18.4	20.5	20.9	13.2	**12.2**
	Std. Dev.	36.2	30.2	**2**	40.6	3.5	5.9



4.3. Comparisons on the solution accuracy

The mean solutions, the best solution and standard deviation (Std. Dev.) of the solutions are listed in Table 4. The best result among those PSOs is indicated by Boldface

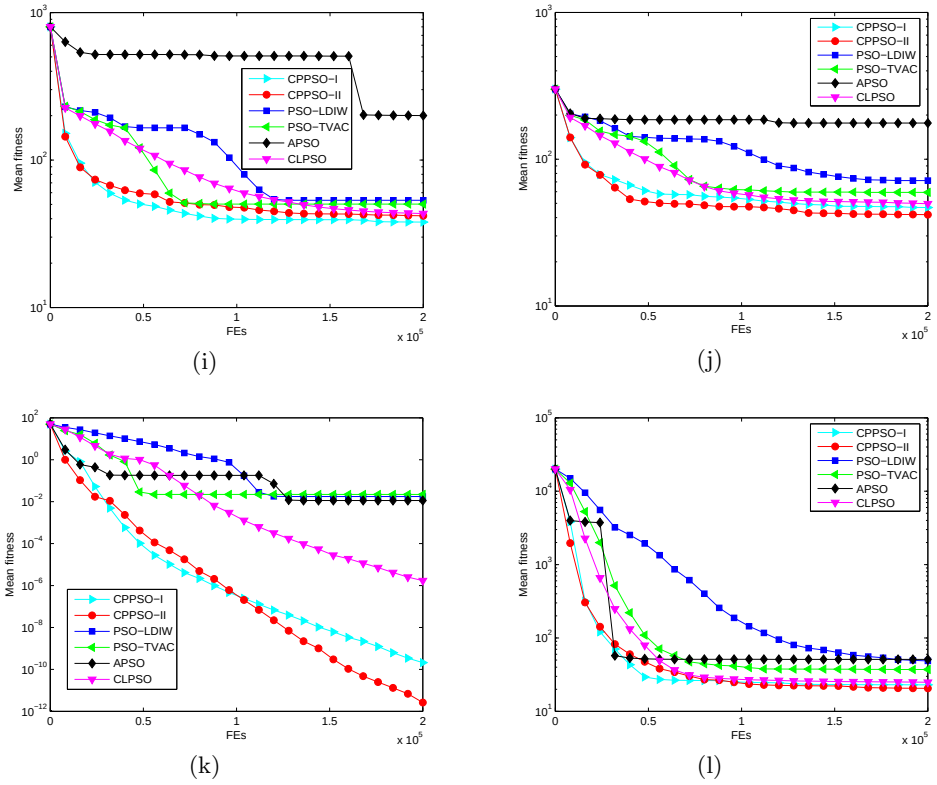


Figure 5: Performance of the algorithms for twelve 30-dimensional benchmark functions.(a) Sphere function. (b) Rosenbrock function. (c) Weierstrass function. (d) Rastrigin function. (e) Noncontinuous Rastrigin function. (f) Ackley function. (g) Griewank function. (h) Generalized Penalized function. (i) Rotated Rastrigin function. (j) Rotated noncontinuous Rastrigin function. (k) Rotated Griewank function. (l) Rotated Rosenbrock function.

in the table. Fig. 5 depicts the comparisons in terms of convergence, mean solutions and evolution processes in solving 12 benchmark functions.

From the Table 4 and Fig. 5(a), it is obviously that, the CPPSO-II provides the best performance on the sphere function, which is used to test the convergent rates. CPPSO-I outperforms CLPSO and ranks the third among these PSOs. Table 4 and Fig. 5(b)-Fig. 5(h) illustrates the comparisons on the multimodal functions without rotation, which are difficult to optimize. CPPSO-I and CPPSO-II achieve the global optimum on the optimization of complex functions f_3, f_4, f_5 , f_7 and f_8. It is worth pointing out that CPPSO-I and CPPSO-II perform much better than other PSOs on f_2, which is usually difficult to optimize. From the Table 4 and Fig. 5(b)- Fig. 5(h), it is found that both CPPSO-I and CPPSO-II have good search performance on multimodal functions. CPPSO-II can refine the search results more accurately than CPPSO-I. Though APSO performs better than CPPSO on f_6, its mean solutions and best solution of other functions are worse than those of the CPPSO-I and CPPSO-II. When tackling rotated functions, CPPSO-I and CPPSO-II surpass CLPSO on four rotated functions, seen from Table 4 and Fig. 5(i)-Fig. 5(l). CPPSO-II offers a little better performance on the rotated problems than CPPSO-I does except f_9. We observe that the method proposed in this paper can help the PSO to search the optimum as well as maintain a high convergence speed. The capability

of the proposed approach to avoid local optima and find global optimum of multimodal functions indicates the superiority of CPPSO.

Comparing the results and the convergence graphs obtained using above mentioned 6 PSO algorithms, CLPSO has good global search ability and slow convergence speed. APSO can converge to the best solution found so far quickly though it is easy to stuck in the local optima. Its search performance is seriously affected after rotation. CPPSO has better local search ability and global search ability although CPPSO's performance is also affected by the rotation.

4.4. Comparisons on convergent rate

The convergent rate for achieving the global optimum is another key point for measuring the algorithm performance. Note that in solving real-world optimization problems, the "FE" overwhelms the algorithm overhead. Generally, the FE accounts for the most time as the PSO is highly computation efficient. Hence, the computational complexities of these algorithms are not compared here. Table 5 shows that CPPSO needs least FEs to achieve the acceptable solution on f_2, f_3, f_5, f_6, f_7, f_8, f_9, revealing that CPPSO has a higher convergent rate than other algorithms do. Though APSO outperforms CPPSO-I and CPPSO-II on the other functions, CPPSO-I or CPPSO-II rank second on these functions. In addition, APSO has much worse successful ratio and accuracy than CPPSO-I and CPPSO-II do on the tested functions.

4.5. Comparisons on successful ratio

Table 5 also shows that CPPSO yields the highest ratio for achieving acceptable solutions in 30 runs. According to the no free lunch theorem [36], any elevated performance over one class of problems is offset by performance over another class. Hence, one algorithm cannot perform better than all others on every problem.

In summary, the CPPSO performs best on both unimodal and multimodal functions. The CPPSO possesses capabilities of fast convergence, highest successful ratio, least FEs and best search accuracy among these PSOs. The performance arises from the competitive and penalized mechanism, adaptive inertia weight and ELLA.

5. Analysis of adaptive inertia weight and controllable probabilistic approach

Two techniques, i.e., adaptive inertia weight and controllable probabilistic approach are also used to test the effects of them on the search performance of CPPSO. The performance of CPPSO-I without adaptive inertia weight or controllable probabilistic approach is tested, respectively. Results of 30 independent runs are shown in Table 6.

It is clear from the results that with both adaptive inertia weight and controllable probabilistic approach, CPPSO-I outperforms other variants of CPPSO-I on search accuracy. CPPSO-I can not only deliver the highest accuracy on unimodal functions, but also deliver a good global search performance on multimodal functions. Moreover, with only controllable probabilistic technique, CPPSO-I still performs well on tested functions when testing search accuracy. However, CPPSO-I with only controllable probabilistic method

Table 5: Convergence speed and algorithm reliability comparisons; '-' representing no runs reached an acceptable solution

		LDIW	TVAC	CLPSO	APSO	CPPSO-I	CPPSO-II
f_1	Mean FEs	106534	45339	56745	**7978**	21735	9385
	Ratio(%)	**100**	**100**	**100**	**100**	**100**	**100**
f_2	Mean FEs	103910	45741	48910	24254	21641	**19248**
	Ratio(%)	**100**	**100**	**100**	**100**	**100**	**100**
f_3	Mean FEs	118981	0	106211	31451	34016	**31022**
	Ratio(%)	**100**	0	**100**	10	**100**	**100**
f_4	Mean FEs	92437	35056	62535	**3418**	9451	4479
	Ratio(%)	**100**	70	**100**	**100**	**100**	**100**
f_5	Mean FEs	101656	47491	47440	3160	8788	**3073**
	Ratio(%)	**100**	83.3	**100**	**100**	**100**	**100**
f_6	Mean FEs	110427	54642	63212	40209	**13801**	15235
	Ratio(%)	**100**	96.7	**100**	**100**	**100**	**100**
f_7	Mean FEs	55331	18136	75658	72629	27448	**22342**
	Ratio(%)	16.7	13.3	**100**	**100**	**100**	**100**
f_8	Mean FEs	66340	36411	56957	27773	16058	**10687**
	Ratio(%)	**100**	80	**100**	**100**	**100**	**100**
f_9	Mean FEs	50820	36411	123109	-	**43551**	79884
	Ratio(%)	50	80	93.3	0	**96.7**	90
f_{10}	Mean FEs	10271	21717	50219.8	**11208**	58179	60782
	Ratio(%)	3.3	30	43.3	10	66.7	**76.7**
f_{11}	Mean FEs	40382	12959	84169	**11545**	30691	24807
	Ratio(%)	36.67	23.3	**100**	66.7	**100**	**100**
f_{12}	Mean FEs	95954	44893	44414.6	**11665.1**	27782	29291
	Ratio(%)	76.7	93.3	**100**	93.3	**100**	**100**
	Mean Reliability	73.6	64.1	94.7	73.3	96.95	**97.2**

needs more FEs to get the acceptable solutions.

On the other hand, the CPPSO with only adaptive inertia weight and without controllable probabilistic approach can obtain acceptable solutions rapidly. However, it suffers from bad search performance for global optimum on the tested functions. It is also worth mentioning that, CPPSO with adaptive inertia weight, combined with controllable probabilistic approach can provide a much better performance on accuracy as well as has a fast convergence speed. It has been shown from Table 6 that adaptive inertia weight can enhance the convergence capability. However, PSO with controllable probabilistic approach alone and without parameter switching offers a good performance for searching global optimum on the functions, as discussed above.

To summarize, the full CPPSO is the most powerful for the tested functions. The results verify that adaptive inertia weight can accelerate the convergence and controllable probabilistic approach can help the swarm to have a better global search ability.

Table 6: Advantages of adaptive inertia weight and controllable probabilistic approach

	Algorithms	CPPSO-I with both adaptive weight and controllable probability	CPPSO-I with controllable probability	CPPSO-I with adaptive weight
f_2	Average	**0.02**	9.8	19.8
	Std. Dev.	**0.0001**	13.5	27.3
	FEs	21641	33462	**11463**
f_4	Average	**0**	2.3×10^{-13}	0.2
	Std. Dev.	**0**	6.5×10^{-13}	0.5
	FEs	**9451**	61816	20384
f_5	Average	**0**	2.6	14.4
	Std. Dev.	**0**	1.4	2.7
	FEs	**8788**	77354	16094
f_6	Average	$\mathbf{2.4}\times10^{-14}$	$\mathbf{2.4}\times10^{-14}$	1.1×10^{-14}
	Std. Dev.	$\mathbf{4.1}\times10^{-15}$	$\mathbf{4.1}\times10^{-15}$	1.2×10^{-15}
	FEs	**13801**	58052	14614
f_7	Average	**0**	**0**	5.3×10^{-12}
	Std. Dev.	**0**	**0**	2.7×10^{-12}
	Mean FEs	27448	50273	**15381**

6. Synchronization of an array of delayed neural networks via a CPPSO

In this section, we will utilize the proposed novel CPPSO to design the controller for synchronization of an array of neural networks with mixed time-delays.

6.1. An array of delayed neural networks model

To facilitate the readers, let us present the complex networks in a step-by-step way. We start with the following master network:

$$ds(t) = (-Cs(t) + Af(s(t)) + Bg(s(t-\tau_1)) + D\int_{k=t-\tau_2}^{t} h(s(k))dk)dt \tag{34}$$

where $s(t) = (s_1(t), s_2(t), \cdots, s_n(t))^T \in \mathbb{R}^n$ is the state vector of the network; A is a constant matrix; matrices B and C are the connection weight matrix and the delayed connection weight matrix, respectively; τ_1 is a time delay and τ_2 is the distributed delay. $f(s(t)) = (f_1(s(t)), \cdots, f_n(s(t)))^T$, $g(s(t)) = (g_1(s(t)), \cdots, g_n(s(t)))^T$ and $h(s(t)) = (h_1(s(t)), \cdots, h_n(s(t)))^T$. In this paper, an array of linearly coupled identical networks with mixed time-delays under study is proposed as follows:

$$\begin{aligned} dx_i(t) = {} & [-Cx_i(t) + Af(x_i(t)) + Bg(x_i(t-\tau_1)) + D\int_{k=t-\tau_2}^{t} h(s(k))dk \\ & + \sum_{j=1}^{N} w_{ij}\Gamma x_j(t) + u_i(t)]dt, i = 1, 2, \cdots, N, \end{aligned} \tag{35}$$

where $x_i(t) = [x_{i1}(t), x_{i2}(t), \cdots, x_{in}(t)]^T \in \mathbb{R}^n$ is the state vector of the ith node; $u_i(\cdot)$ is the control input to ensure that $x_i(t) - s(t) \to 0$ as $t \to \infty$; Γ represents the inner coupling matrix between the subsystems; $W = (w_{ij})_{N\times N}$ is the coupling configuration matrices representing the coupling strength and the topological structure of the networks, satisfying

$$w_{ii} = -\sum_{j=1, j\neq i}^{N} w_{ij}, i, j = 1, 2, \cdots, N. \tag{36}$$

In order to investigate the synchronization for coupled networks (35), we let $e_i(t) = x_i(t) - s(t)$ be the synchronization error. Then, the error system follows immediately from (34) and (35) as follows:

$$\begin{aligned} de_i(t+1) &= [-Ce_i(t) + A\tilde{f}(e_i(t)) + B\tilde{g}(e_i(t-\tau_1)) + D\int_{k=t-\tau_2}^{t} \tilde{h}(s(k))dk \\ &\quad + \sum_{j=1}^{N} w_{ij}\Gamma e_j(t) + u_i(t)]dt, i = 1, 2, \cdots, N, \end{aligned} \tag{37}$$

where $\tilde{f}(e_i(t)) = f(x_i(t)) - f(s(t))$, $\tilde{g}(e_i(t)) = g(x_i(t)) - g(s(t))$ and $\tilde{h}(e_i(t)) = h(x_i(t)) - h(s(t))$.

We are going to design a controller $u_i(t)$ in order to make the coupled system (37) synchronized. For simplicity of the implementation, we adopt the following memoryless state-feedback controller:

$$u_i(t) = Ke_i(t). \tag{38}$$

Substitute (38) into (37) to give the following closed-loop system:

$$\begin{aligned} de_i(t+1) &= [(-C + K)e_i(t) + A\tilde{f}(e_i(t)) + B\tilde{g}(e_i(t-\tau_1)) + D\sum_{m=-\tau_2}^{-1} \tilde{h}(e_i(k))dk) \\ &\quad + \sum_{j=1}^{N} w_{ij}\Gamma e_j(t)]dt, i = 1, 2, \cdots, N. \end{aligned} \tag{39}$$

The problem of synchronization is similar and can be viewed as follows:

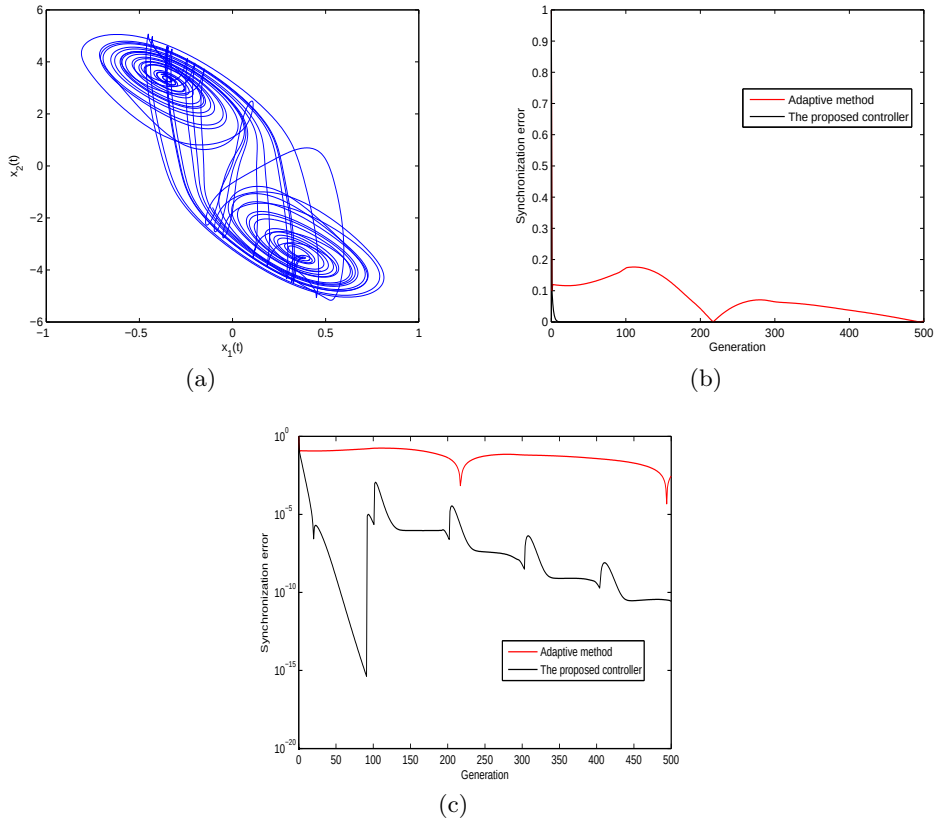
$$J = \sum_{k=1}^{M}\sum_{i=1}^{N} \|e_i(k)\|^2, i = 1, 2, \cdots, N, \tag{40}$$

where $k = 1, 2, \cdots, M$ is the sampling time point and M denotes the length of data used for synchronization. The synchronization of system (39) can be achieved by searching suitable K^* such that the objective function (40) is minimized, i.e.

$$(K^*) = \arg\min_{(K)\in\Omega}(J), \tag{41}$$

where Ω is searching space admitted for control gain.

Remark 6. Recently, synchronization problems of an array of continuous-time delayed neural networks have been investigated extensively in [4, 5, 15, 25, 26, 38]. The controller designed in above works are used in terms of LMIs or adaptive controller. However, the results derived in the works need a number of assumptions and have conservativeness. In this paper, we aim to deal with synchronization problem of an array of neural networks using a CPPSO. It can be seen that the synchronization of an array of neural networks is achieved with high efficiency.



(a) (b)

(c)

Figure 6: Time evolution of synchronization error.

Consider the following single node of complex networks (34) with the following parameters:

$$C = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, A = \begin{pmatrix} 1.8 & -0.15 \\ -5.2 & 3.5 \end{pmatrix}, B = \begin{pmatrix} -1.7 & -0.12 \\ -0.26 & -2.5 \end{pmatrix}, D = \begin{pmatrix} 0.6 & 0.15 \\ -2 & -0.1 \end{pmatrix},$$

$\tau_1 = 1, \tau_2 = 0.9$. The synchronization error is computed as

$$E(k) = \frac{1}{N}\sqrt{\sum_{i=1}^{N} e_i^2(k)}, \tag{42}$$

The topology and inner coupling matrix of an array of neural networks with mixed time-delays are considered as follows:

$$G = \begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix}, \Gamma = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Like adaptive controller [25, 26], supposing that the controller is designed as follows:

$$K = \begin{pmatrix} K_{11} & 0 \\ 0 & K_{22} \end{pmatrix}. \tag{43}$$

We use CPPSO-I to design the controller of an array of neural networks. The population size of CPPSO-I is set 10 and the generation for PSO is 20. The Runge-Kutta method is used to solve the the (34) and (35). The generation number of neural networks is set 500 and the step size is 0.01 second. Fig. 6(a) shows chaotic behaviors of the neural networks (34). Fig. 6(b) shows the synchronization error and Fig. 6(c) illustrates the synchronization error in logarithm form. It can be seen from Fig. 6(b) and Fig. 6(c) that the error states of synchronization tend to zero faster and more accurate than the conventional adaptive method in [25, 26], which demonstrates the great efficiency of CPPSO presented in this paper.

7. Conclusion

The controller design problem has been investigated for synchronization of an array of neural networks with mixed time-delays via a CPPSO algorithm. Based on the information of fitness value, an evolutionary state function is defined and computed, which offers an effective way to control inertia weight. As illustrated in the benchmark tests, the adaptive control of the inertia weight enables the PSO more efficient, providing an improved convergence speed in terms of FEs to reach acceptable solutions for benchmark functions.

A velocity updating equation with Bernoulli stochastic variables is proposed to make the particles learn from different strategies. The learning strategies are automatically selected efficiently using a competitive penalized method. Furthermore, an elite local learning approach is developed to lead the swarm to refine converging solutions efficiently. The searching radius is switched between different values governed by a Markov chain in ELLA. The substantially improved global solution accuracy as a result of the ESF, PSO with controllable probability and ELLA are verified in both unimodal and multimodal problems.

Note that the ESF, PSO with controllable probability and ELLA are easy to set and require no burden to implement. Therefore, the CPPSO is simple and easy to use as the standard PSO, since it offers in substantially improved performance in terms of convergence speed and solution accuracy.

Finally, we have employed the CPPSO to design the memoryless feedback controller for synchronization of an array of delayed neural networks. Compared with the results obtained by Lyapunov method, the controller design using CPPSO has faster convergence speed.

We would like to point out that it is possible to extend our main results to design control scheme of acceleration coefficients using ESF and apply our methods to other evolutionary computation algorithms. The corresponding results will appear in our future works.

8. Acknowledgement

The authors are grateful to the Editor and anonymous reviewers for their careful reading and constructive comments.

References

[1] P. S. Andrews, An investigation into mutation operators for particle swarm optimization, in Proc. IEEE Congr. Evol. Comput., Vancouver, BC, Canada, 2006, pp. 1044-1051.

[2] P. J. Angeline, Using selection to improve particle swarm optimization, in Proc. IEEE Congr. Evol. Comput., Anchorage, AK, 1998, pp. 84-89.

[3] F. VD. Bergh and A. P. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE Trans. Evol. Comput. , vol. 8.,no. 3 pp.225-239, June 2004.

[4] J. Cao, P. Li, and W. W. Wang, Global synchronization in arrays of delayed neural networks with constant and delayed coupling, Phys. Lett. A, vol. 353, pp. 318-325, 2006.

[5] G. Chen, J. Zhou, and Z. R. Liu, Global synchronization of coupled delayed neural networks and applications to chaotic CNN models, Int. J. Bifur. Chaos, vol. 14, no. 7, pp. 2229-2240, 2004.

[6] Y. P. Chen, W. C. Peng, and M. C. Jian, Particle swarm optimization with recombination and dynamic linkage discovery, IEEE Trans. Syst., Man, Cybern. B, Cybern., vol. 37, no. 6, pp. 1460-1470, Dec. 2007.

[7] M. Clerc, J. Kennedy, The particle swarm: explosion, stability, and convergence in a multi-dimensional complex space, IEEE Transactions on Evolutionary Computation 6(1). Piscat-away, NJ, 2002, pp. 58-73.

[8] R. C. Eberhart and Y. H. SHi, Particle swarm optimization: Developments, applications and resouces, in Proc. IEEE Congr. Evol. Comput. Seoul, Korea, 2001, pp. 81-86.

[9] H. Gao, J. Lam, and G. Chen, New criteria for synchronization stability of general complex dynamical networks with coupling delays, Phys. Lett. A, vol. 360, pp. 263-273, 2006.

[10] C. M. Gray, Synchronous oscillations in neuronal systems: Mechanism and functions, J. Comput. Neurosci., vol. 1, pp. 11-38, 1994.

[11] R. Mendes, J. Kennedy, and J. Neves, The fully informed particle swarm: Simpler, maybe better, IEEE Trans. Evol. Comput., vol. 8, no. 3, pp. 204-210, Jun. 2004.

[12] J.J. Liang, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evol. Comput., vol. 10, no. 3, pp. 281-295, Jun. 2006.

[13] J. Liang, Z. Wang and X. Liu, Robust synchronization of an array of coupled stochastic discrete-time delayed neural networks, IEEE Tran. on Neural Networks, 9(2008)1910-1921.

[14] J. Liang, Z. Wang, X. Liu, Robust passivity and passification of stochastic fuzzy time-delay systems Information Sciences, 180(2010), 1725-1737.

[15] W. L. Lu and T. P. Chen, Synchronization of coupled connected neural networks with delays, IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 51, no. 12, pp. 2491-2503, Dec. 2004.

[16] W. L. Lu and T. P. Chen, Synchronization analysis of linearly coupled networks of discrete time systems, Physica D, vol. 198, pp. 148-168, 2004.

[17] J. H. Lü, X. H. Yu, and G. Chen, Chaos synchronization of general complex dynamical network, Physica A, vol. 334, pp. 281-302, 2004.

[18] Y. Wang, Y. Yang, Particle swarm optimization with preference order ranking for multi-objective optimization, Information Sciences, 179(2009)1944-1959.

[19] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference On Neural Network, 1995, pp. 1942-1948.

[20] R.A.Krohling and L.dos Santos Coelho, Coevolutionary particle swarm optmization using Gaussian distribution for solving constrned optimization problems, IEEE Trans. Syst., Man, Cybern. B, vol.36, no.6, pp.1407-1416, Dec. 2006.

[21] L. M. Pecora and T. L. Carroll, Synchronization in chaotic systems, Phys. Rev. Lett., vol. 64, no. 8, pp. 821-824, 1990.

[22] A. Ratnaweera, SK. Halgamure, HC. Watson. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Trans Evol. Comput. 2004;8:240-55.

[23] Y. Shi, RC. Eberhart. Empirical study of particle swarm optimization. In: Proceedings of the 1999 IEEE congress on evolutionary computation. Piscataway (NJ): IEEE Press; 1999. p. 1945-50.

[24] Y. Shi, RC Eberhart. Parameter selection in particle swarm optimization. In: Proceedings of the 7th international conference on evolutionary programming VII. LNCS, vol. 1447. New York: Springer-Verlag; 1998. p. 591-600.

[25] Y. Tang, R. Qiu, J. Fang, Q. Miao, M. Xia, Adaptive lag synchronization in unknown stochastic chaotic neural networks with discrete and distributed time-varying delays, Physics letters A, 372(2008)4425-4433.

[26] Y. Tang, J. Fang, Robust synchronization in an array of fuzzy delayed cellular neural networks with stochastically hybrid coupling, Neurocomputing 72 (2009) 3253-3262.

[27] Y. Tang, Z. D. Wang and J. Fang, Pinning control of fractional-order weighted complex networks, Chaos,19(2009)013112.

[28] Y. Tang, J. Fang, M. Xia, D. Yu, Delay-distribution-dependent stability of stochastic discrete-time neural networks with randomly mixed time-varying delays, Neurocomputing, 72 (2009) 3830-3838.

[29] R. Salomon, Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions, BioSystems, vol. 39, pp. 263-278, 1996.

[30] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger and S. Tiwari, Problem definitions and evaluation criteria for the CEC2005 special session on real-parameter optimization, in Proc. IEEE Congr. Evol. Comput.,2005, pp.1-50.

[31] Z. Wang, F. Yang, Daniel W. C. Ho, and X. Liu, Robust H_∞ Control for Networked Systems With Random Packet Losses, IEEE Trans Systems, man and cybernetics-PART B, VOL. 37, NO. 4, AUGUST 2007.

[32] Z. Wang, F. Yang, D.W. C. Ho, and X. Liu, Robust finite-horizon filtering for stochastic systems with missing measurements, IEEE Signal Process. Lett., vol. 12, no. 6, pp. 437-440, Jun. 2005.

[33] Z. Wang, D. W. C. Ho., and X. Liu. Variance-constrained filtering for uncertain stochastic systems with missing measurements. IEEE Transactions on Automatic Control, 48(2003)1254-1258.

[34] C. W. Wu, Synchronization in arrays of coupled nonlinear systems with delay and nonreciprocal time-varying coupling, IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 52, no. 5, pp. 282-286, May 2005.

[35] A. A. Ukhtomsky, Collected Works (in Russian). Leningrad, Russia: Nauka, 1978, pp. 107-237.

[36] D. H. Wolpert and W. G. Macready, No free lunch theorems for optimization, IEEE Congr. Evol. Comput. vol. 1, no, 1, pp. 67-82, Apr. 1997.

[37] X. Yao, Y. Liu and G. M. Lin, Evolutionary programming made faster, IEEE Trans. Evol. Comput., vol.3, no.2, pp.82-102, July,. 1999.

[38] W. Yu, J. Cao, J. Lü, Global Synchronization of Linearly Hybrid Coupled Networks with Time-Varying Delay, SIAM J. applied dynamical systems, 7(2008)108-133.

[39] Z. Zhan, J. Zhang, Y. Li, H.S.H. Chung, Adaptive particle swarm optimization, IEEE Trans. System, man and cybernetics-B, 39(2009)1362-1381.